

sgClaw 项目团队分工

总览

角色	负责人	技术栈	工作量估算	产出物
P1a 核心通信	赵义仑	Rust, ZeroClaw	~900 行代码	管道通信 + 浏览器工具
P1b 业务支持	TBD	Rust, ZeroClaw	~600 行代码	Skill 加载 + 记忆管理
P2 浏览器对接	TBD	C++, Chromium	~600 行代码	浏览器端桥接模块
P3 业务技能	TBD	JavaScript, 提示词工程	AI 辅助迁移 400+ 场景	Skill 仓库管理
P4 前端发布	TBD	Vue, DevOps	~150 行 + 工具链	UI 面板 + Skill 后台 + 打包

总代码量：约 2250 行（自研部分），基于 ZeroClaw 框架 **84% 复用** 团队规模：5 人 交付周期：2 周

P1a · 核心通信开发（关键路径）

角色定位：Rust 全栈工程师，负责 sgClaw 与浏览器的核心通信链路 **负责人：**赵义仑

主要职责

1. 管道通信层 (Pipe Protocol)

- 实现 STDIO Pipe 双向通信协议
- JSON Line 编解码
- 序列号 (sequence_id) 递增校验
- 握手协议和版本校验
- 消息大小限制 (≤1MB)

2. 浏览器操作工具 (BrowserPipeTool)

- 实现 ZeroClaw 的 `Tool` trait
- 封装 15 个浏览器 Action：
 - 导航：navigate, reload, goBack, goForward
 - 交互：click, type, select, scrollTo
 - 读取：getText, getHtml, getAomSnapshot, pageScreenshot
 - 等待：waitForSelector, waitForNavigation
 - 存储：storageSet, storageGet
- 错误处理和重试机制

3. 安全策略层 (MAC Policy)

- 实现域名白名单校验
- Action 白名单校验
- Human-in-the-loop 确认机制
- 熔断器（连续失败 > 10 次自动停止）

- HMAC 签名校验

技术栈

- 语言: Rust (edition 2021)
- 框架: ZeroClaw 0.x
- 异步: tokio
- 序列化: serde, serde_json
- 加密: ed25519-dalek, hmac, sha2

产出物

```
sgClaw/  
src/  
  main.rs          # 入口, 管道初始化  
  pipe/  
    protocol.rs    # Pipe 协议定义  
    reader.rs      # STDIN 读取  
    writer.rs      # STDOUT 写入  
  tool/  
    browser_pipe.rs # BrowserPipeTool (Tool trait impl)  
  security/  
    mac_policy.rs  # MAC 白名单策略  
    hmac.rs        # HMAC 校验  
  config/  
    settings.rs    # 配置文件解析
```

代码量: 约 900 行 (不含 ZeroClaw 框架复用代码)

关键接口

```
// ZeroClaw 定义的 Tool trait  
pub trait Tool {  
    async fn execute(&self, input: &str) -> Result<String>;  
}  
  
// sgClaw 实现  
pub struct BrowserPipeTool {  
    pipe_writer: PipeWriter,  
    mac_policy: MacPolicy,  
    sequence_id: AtomicU64,  
}  
  
impl Tool for BrowserPipeTool {  
    async fn execute(&self, input: &str) -> Result<String> {  
        // 1. 解析 JSON 命令  
        // 2. MAC 校验  
        // 3. 写入 Pipe  
        // 4. 等待响应
```

```
// 5. 返回结果
    }
}
```

里程碑

时间	任务	验收标准
Day 1-2	Pipe 通信打通	STDIN/STDOUT 双向收发 JSON
Day 3-4	BrowserPipeTool 实现	完成 click/type/navigate 三个 Action
Day 5	W1 里程碑	LLM → Pipe → Browser 链路全通
Day 6-7	MAC 策略集成	白名单校验生效，人工确认流程通
Day 8-9	联调测试	与 P1b、P2 联调，与 P3 联调 Skill 加载
Day 10	W2 里程碑	交付完整通信模块

P1b · 业务支持开发

角色定位：Rust 工程师，负责 Skill 系统和记忆管理

主要职责

1. Skill 加载器 (SkillLoader)
- 扫描 Skill 目录，解析元数据

◦ Ed25519 签名验证

◦ JavaScript 沙箱执行环境

◦ Skill 注册到 Agent 工具列表
2. 记忆管理 (CompositeMemory)
- L0 即时记忆 (LLM Context)

◦ L1 短期记忆 (Ring Buffer, 50 条 / 8000 tokens)

◦ L2 长期记忆 (SQLite + 向量索引)

◦ 记忆检索和压缩
3. 评估器 (Critic)
- 质量评估逻辑 (成功 / 失败 / 需重试)

◦ 失败计数和熔断触发

◦ 执行日志记录
4. 主循环 (AgentRuntime)
- 集成 ZeroClaw 的 ReAct Loop

◦ 协调各模块运行

◦ 异步任务调度 (tokio)

技术栈

- 语言： Rust (edition 2021)
- 框架： ZeroClaw 0.x
- 异步： tokio
- 数据库： rusqlite (SQLite)
- 向量： faiss-rs 或 hnswlib （可选）
- JS 运行时： rusty_v8 或 boa_engine

产出物

```
sgClaw/  
  src/  
    agent/  
      runtime.rs      # AgentRuntime , ReAct 循环  
      critic.rs       # 评估器  
    skill/  
      loader.rs       # Skill 发现与加载  
      executor.rs     # JavaScript 沙箱执行  
      signature.rs    # Ed25519 签名验证  
    memory/  
      composite.rs    # CompositeMemory (Memory trait impl)  
      ring_buffer.rs  # L1 短期记忆  
      sqlite_store.rs # L2 长期记忆  
    llm/  
      provider.rs     # LLM Provider 抽象 (ZeroClaw)
```

代码量：约 600 行（不含 ZeroClaw 框架复用代码）

协作依赖

- 依赖 P1a：等 Day 5 后 Pipe 通信链路打通，再集成 AgentRuntime
- 支持 P3：提供 Skill 签名验证和加载能力，Day 6-7 联调
- 支持 P4：Memory 模块的 SQLite 操作可由 P4 协助（如有余力）

里程碑

时间	任务	验收标准
Day 1-3	SkillLoader 开发	能扫描、解析、签名验证 Skill
Day 4-5	Memory 开发	Ring Buffer + SQLite 存取正常
Day 6-7	AgentRuntime 集成	集成 P1a 的 BrowserPipeTool
Day 8-9	Critic + 测试	评估器生效，联调测试
Day 10	W2 里程碑	交付完整 Agent 引擎

P2 · 浏览器对接

角色定位: C++ Chromium 工程师，负责浏览器端桥接

主要职责

1. 进程宿主 (SgClawProcessHost)
- Singleton 单例模式
 - 启动 sgclaw Rust 二进制进程
 - 使用 `base::LaunchProcess` 创建子进程并传递 STDIO Pipe fd/HANDLE
 - 进程状态管理: Idle → Starting → Running → Stopping → Stopped / Crashed
 - 崩溃检测与自动重启 (可选)
 - 优雅停止 (发送 SIGTERM / CloseHandle)
2. 管道监听器 (PipeListener)
- 监听 STDOUT fd, 异步读取 JSON Line
 - 解析 JSON 命令 (action + params)
 - 消息大小检查 (≤1MB)
 - 分发给 CommandRouter
3. MAC 白名单检查 (MacWhitelistCheck)
- 读取 `rules.json` 配置文件
 - 检查域名白名单 (当前 Tab URL 是否在允许列表)
 - 检查 Action 白名单 (是否允许执行该操作)
 - 返回: `Allow` | `NeedConfirm` | `Deny`
4. 与现有系统集成
- **CommandRouter** (已有 70+ 命令): 零修改, sgClaw 只是一个新的命令来源
 - **FunctionsUI**: 新增 Side Panel UI 入口, 调用 sgClaw API
 - **CdpBridgeManager / ZombiePageManager**: 复用现有能力, 无需修改
 - **RpaGlobalStorage**: 复用存储接口

技术栈

- **语言:** C++17
- **框架:** Chromium base library
- **IPC:** STDIO Pipe (fd / HANDLE)
- **JSON:** `base::JSONReader` / `base::JSONWriter`
- **线程:** `base::SequencedTaskRunner`

产出物

```
src/chrome/browser/superrpa/sgclaw/
sgclaw_process_host.h           # 进程宿主头文件
sgclaw_process_host.cc          # 实现 (~200-300 行)
pipe_listener.h                 # 管道监听器头文件
pipe_listener.cc                 # 实现 (~150 行)
mac_whitelist_check.h           # MAC 检查头文件
```

```
mac_whitelist_check.cc      # 实现 (~100 行)
rules.json                  # 白名单配置
BUILD.gn                    # 编译规则

src/chrome/browser/ui/views/side_panel/sgclaw/
sgclaw_panel_view.h         # Side Panel UI (可选, P4 主导)
sgclaw_panel_view.cc
```

代码量: 约 500-600 行 C++ (新增), 现有代码零修改

关键接口

```
// SgClawProcessHost.h
class SgClawProcessHost {
public:
    static SgClawProcessHost* GetInstance();

    void Start();
    void Stop();

    enum class ProcessState {
        kIdle,
        kStarting,
        kRunning,
        kStopping,
        kStopped,
        kCrashed
    };
    ProcessState GetState() const;

private:
    SgClawProcessHost();
    base::Process process_;
    std::unique_ptr<PipeListener> listener_;
    ProcessState state_;
};

// PipeListener.h
class PipeListener {
public:
    explicit PipeListener(base::ScopedFD stdout_fd);
    void StartListening();

    using MessageCallback = base::RepeatingCallback<void(const
base::Value&)>;
    void SetMessageCallback(MessageCallback callback);

private:
    void ReadLoop();
    base::ScopedFD fd_;
    MessageCallback callback_;
```

```
};

// MacWhitelistCheck.h
enum class MacResult { kAllow, kNeedConfirm, kDeny };

class MacWhitelistCheck {
public:
    static MacWhitelistCheck* GetInstance();
    MacResult Check(const std::string& domain, const std::string& action);

private:
    void LoadRules();
    std::set<std::string> allowed_domains_;
    std::set<std::string> allowed_actions_;
};
```

与现有系统关系



零侵入设计: sgClaw 不在时, 浏览器完全正常工作

里程碑

时间	任务	验收标准
Day 1-2	ProcessHost 基础框架	能启动 dummy 进程
Day 3-4	Pipe 双向通信	能收发 JSON 消息
Day 5	W1 里程碑	与 P1a 联调成功
Day 6-7	MAC 白名单 + CommandRouter 对接	安全策略生效
Day 8-9	集成测试	浏览器端稳定运行

P3 · 业务技能开发

角色定位: 业务 + AI 工程师, 负责 400+ 场景迁移和 Skill 仓库管理

主要职责

1. 黄金样本制作 (Day 1-3)

- 精选 10-15 个代表性场景
- 覆盖各种模式：表单填写、审批流、数据采集、跨系统同步、定时巡检
- 手工编写为**标准 Skill 格式** + 详细注释
- 作为 Few-shot 范例

2. 提示词工程 (Day 3-4)

- 设计 System Prompt: agent-vue → sgClaw Skill 翻译专家
- 在 10 个样本上测试，迭代到 **90%+ 准确率**
- 约束：只能用沙箱 API，禁止 fetch/require/eval

3. 批量 AI 翻译 (Day 5-7)

- 用 **Qwen-72B** 或 **DeepSeek** 批量处理 400+ 场景
- 每批 20-30 个场景
- 输出：400+ 个 Skill 候选文件
- 自动生成元数据 + 参数 Schema

4. 自动化测试与质量把控 (Day 7-9)

- 静态检查：元数据完整性、签名格式、参数 Schema
- 语法检查：JS 语法、沙箱 API 合规性
- Mock 执行：不连真实系统，只测结构
- **人工抽检**：每 20 个抽 1 个深度测试

5. 业务验证 (Day 10+)

- 交给业务方按场景分批验证
- 错误反馈 → 人工修正 → 更新提示词
- 持续自进化

6. Skill 仓库管理

- 建立 Skill 目录结构
- 编写 Skill 开发文档
- 制作签名工具脚本
- Skill 版本管理

技术栈

- **语言**: JavaScript (ES6+)
- **AI**: 内网大模型 (Qwen-72B / DeepSeek)
- **提示词工程**: Few-shot learning, System Prompt 设计
- **业务系统**: OA、ERP、财务、HR 等
- **工具**: Node.js (签名工具), JSON Schema
- **加密**: Ed25519 (通过 Node.js crypto)

Skill 定义格式


```
/**
 * @skill    erp-monthly-report
 * @version  1.0.0
 * @author   P3
 * @domains  erp.example.com
 * @params   {
 *   "month": { "type": "string", "pattern": "^\\d{4}-\\d{2}$" },
 *   "format": { "type": "string", "enum": ["pdf", "excel"] }
 * }
 * @description 导出 ERP 月度合规报表
 * @signature <ed25519_signature_base64>
 */

async function execute(params, browserAction) {
  // 1. 导航到 ERP 报表页
  await browserAction('navigate', {
    url: 'https://erp.example.com/report'
  });

  // 2. 等待页面加载
  await browserAction('waitForSelector', {
    selector: '#report-form',
    timeout: 5000
  });

  // 3. 填写月份
  await browserAction('type', {
    selector: '#month-input',
    text: params.month
  });

  // 4. 选择格式
  await browserAction('select', {
    selector: '#format-select',
    value: params.format
  });

  // 5. 点击导出
  await browserAction('click', {
    selector: '#export-button'
  });

  // 6. 等待下载完成
  await browserAction('waitForSelector', {
    selector: '.success-message',
    timeout: 30000
  });

  return { success: true, message: '报表已导出' };
}
```

```
sgClaw/skills/
  built-in/
    黄金样本/                                # 10-15 个手工精制
      erp-monthly-report.js
      oa-batch-approve.js
      ...
    AI生成/                                # 400+ 个 AI 翻译
      财务合规/
      OA审批/
      风险监测/
      人资社保/
      营销数据/
      跨系统同步/
      ...
  prompts/
    translation-system.txt                  # System Prompt
    few-shot-examples.json                 # Few-shot 范例
  tools/
    sign-skill.js                         # 签名工具脚本
    verify-skill.js                       # 验证工具
    batch-translate.js                    # 批量翻译脚本
  schema/
    skill-metadata.json                   # 元数据 JSON Schema
  README.md                               # Skill 开发指南
```

工作量：10-15 个黄金样本（手工）+ 400+ 个 AI 生成 + 提示词工程

沙箱约束

允许使用：

- `browserAction(action, params)` — 调用浏览器操作
- `console.log()` — 日志输出
- `JSON.*` — JSON 处理
- `Promise/async/await` — 异步操作
- `setTimeout/setInterval` — 定时器

禁止使用：

- `fetch()` / `XMLHttpRequest` — 网络请求
- `require()` / `import()` — 模块加载
- `process.*` — 进程操作
- `fs.*` — 文件系统
- `eval()` / `Function()` — 动态代码执行

AI 辅助迁移 workflow

阶段	方式	工作量	输出
Phase 1 · 黄金样本	人工精制	Day 1-3	10-15 个代表性 Skill

阶段	方式	工作量	输出
Phase 2 · 提示词工程	迭代优化 Prompt	Day 3-4	翻译准确率 >90%
Phase 3 · 批量翻译	内网大模型处理	Day 5-7	400+ Skill 候选
Phase 4 · 质量把控	自动检查 + 抽检	Day 7-9	通过率 >85%
Phase 5 · 业务验证	分批上线	Day 10+	持续迭代

里程碑

时间	任务	验收标准
Day 1-3	黄金样本制作	10-15 个手工 Skill 完成
Day 4	提示词工程	翻译准确率 >90%
Day 5-7	批量翻译	400+ Skill 文件生成
Day 8-9	质量把控	自动检查通过率 >85%
Day 10	W2 里程碑	交付 Skill 仓库 + 文档

P4 · 前端与发布

角色定位：前端工程师 + DevOps，负责 UI、Skill 后台和打包发布

主要职责

1. Side Panel UI 开发

- Vue 2.6 + Element UI
- AI 助手对话界面
- 指令输入框
- 执行进度显示
- 日志输出面板
- 启停按钮

2. Skill 管理后台（新增）

- Skill 列表 CRUD 界面
- 管理 400+ Skill 的启用/禁用
- Skill 签名验证状态显示
- Skill 执行统计和日志查看
- 元数据编辑和版本管理

3. 与浏览器集成

- 通过 FunctionsUI IPC 与 C++ 通信
- 调用 SgClawProcessHost 接口（Start / Stop）
- 接收执行状态更新

4. 测试框架搭建

- 单元测试 (Jest)
- 集成测试 (Puppeteer / Playwright)
- E2E 场景测试 (6 大业务场景)
- 测试报告生成

5. CI/CD 流水线

- Rust 编译脚本 (Linux + Windows)
- C++ 编译集成到现有 Chromium 构建
- 自动化测试执行
- 打包生成安装包

6. 双平台打包

- **银河麒麟 V10 SP1**: .deb 包
- **Windows 10/11**: .exe 安装包
- 版本号管理
- 发布说明文档

7. 协助 P1b (可选)

- Memory 模块的 SQLite 数据库操作 (如有余力)

技术栈

- **前端**: Vue 2.6, Element UI, JavaScript
- **测试**: Jest, Puppeteer
- **构建**: Node.js, Bash, Python
- **打包**: dpkg-deb (Linux), NSIS (Windows)
- **CI/CD**: GitLab CI / Jenkins

产出物

```
agent-vue/src/view/sgclaw/
  AgentControlPanel.vue      # AI 助手面板 (~100 行)
  SkillManager.vue           # Skill 管理后台 (~150 行)

sgClaw/
  tests/
    unit/                    # Rust 单元测试
    integration/             # Rust + C++ 集成测试
    e2e/                     # E2E 场景测试
      财务合规.spec.js
      OA审批.spec.js
      ...
  scripts/
    build-linux.sh           # Linux 编译脚本
    build-windows.sh         # Windows 交叉编译
    build-chrome.sh          # Chromium 编译
```

```
package-deb.sh           # 打包 .deb
package-exe.sh          # 打包 .exe
.gitlab-ci.yml           # CI/CD 配置
Dockerfile               # 构建环境镜像（可选）

release/
sgclaw-v1.0.0-kylin-v10-amd64.deb
sgclaw-v1.0.0-windows-x64.exe
CHANGELOG.md
INSTALL.md
```

代码量: 约 150 行 Vue (UI + Skill 后台) + 若干脚本

UI 设计

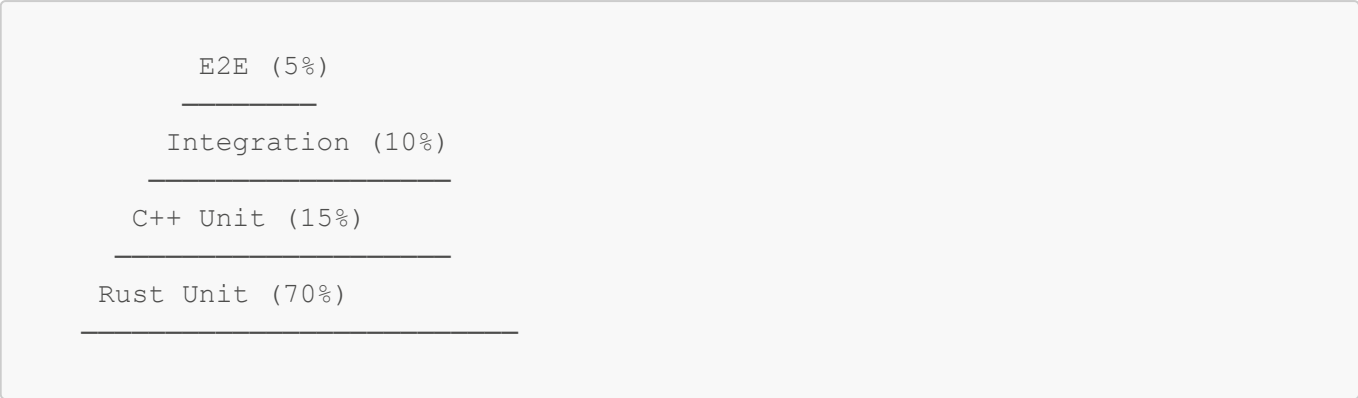
AI 助手面板



Skill 管理后台



测试金字塔



- **70% Rust 单元测试**: mock LLM + mock Pipe
- **15% C++ 单元测试**: ProcessHost / Listener / MAC
- **10% 集成测试**: 真实 sgclaw 进程 + mock Browser
- **5% E2E 测试**: 完整场景 + mock LLM

部署目标

平台	操作系统	架构	包格式	大小
主要	银河麒麟 V10 SP1	x86_64	.deb	~456 MB (浏览器 447MB + sgClaw 9MB)
次要	Windows 10/11	x86_64	.exe	~460 MB

里程碑

时间	任务	验收标准
Day 1-3	Side Panel UI + Skill 后台	界面可用，能发送指令和管理 Skill
Day 4-5	测试框架搭建	单元测试覆盖率 > 70%
Day 6-7	编译脚本 + CI/CD	自动化编译成功
Day 8-9	打包测试	两平台安装包生成
Day 10	W2 里程碑	正式发布 v1.0.0

协作接口

P1a ↔ P2 接口

Pipe 协议规范 (JSON Line over STDIO)

Request (C++ → Rust, via STDIN):

```
{
  "sequence_id": 1,
  "action": "click",
  "params": {
```

```
    "selector": "#submit-button",
    "button": "left"
  },
  "timestamp": 1709499600000
}
```

Response (Rust → C++, via STDOUT):

```
{
  "sequence_id": 1,
  "status": "success",
  "result": {
    "clicked": true
  },
  "error": null,
  "timestamp": 1709499601000
}
```

P1a ↔ P1b 接口

模块依赖关系:

- P1b 的 `AgentRuntime` 依赖 P1a 的 `BrowserPipeTool`
- P1b 的 `SkillLoader` 提供 Skill 列表给 `AgentRuntime`
- P1a 的 `MacPolicy` 在 `BrowserPipeTool` 内部调用

接口约定:

```
// P1a 提供
pub struct BrowserPipeTool;
impl Tool for BrowserPipeTool { ... }

// P1b 使用
let browser_tool = BrowserPipeTool::new(...);
agent_runtime.register_tool(Box::new(browser_tool));
```

P2 ↔ P4 接口

FunctionsUI IPC (Vue ↔ C++)

```
// Vue 调用 C++ 接口
window.superrpa.sgclaw.start()
window.superrpa.sgclaw.stop()
window.superrpa.sgclaw.sendCommand(text)
window.superrpa.sgclaw.onStatusChange((status) => { ... })
window.superrpa.sgclaw.onLog((log) => { ... })
```

```
// Skill 管理接口
window.superropa.sgclaw.listSkills()
window.superropa.sgclaw.toggleSkill(skillId, enabled)
window.superropa.sgclaw.getSkillStats(skillId)
```

P1b ↔ P3 接口

Skill 元数据规范

- 元数据头必须是 JSDoc 注释格式
- @skill 唯一标识符（必需）
- @version 语义化版本号（必需）
- @params JSON Schema（可选）
- @signature Ed25519 签名（必需）

BrowserAction API（Skill 调用）

```
await browserAction(action, params)
// 返回 Promise<result>
```

错误码分类

类别	前缀	示例	责任方
管道层	PIPE_*	PIPE_BROKEN, PIPE_TIMEOUT	P1a, P2
安全层	MAC_*	MAC_DENIED, MAC_INVALID_DOMAIN	P1a, P2
命令层	CMD_*	CMD_INVALID_ACTION, CMD_TIMEOUT	P1a
会话层	SESSION_*	SESSION_EXPIRED	P1b
技能层	SKILL_*	SKILL_SIGNATURE_INVALID, SKILL_NOT_FOUND	P1b, P3
内部层	INTERNAL_*	INTERNAL_ERROR	所有

代码复用分析

ZeroClaw 提供（84%）

- ReAct Loop 循环引擎
- Tool / Provider / Memory trait 定义
- MCP Client (rmcp)
- Streaming Output
- 多 Provider 支持（Claude / GPT / Ollama）
- 配置文件解析
- 日志系统

sgClaw 自研（16%）

模块	行数	负责人
P1a 核心通信		
Pipe Protocol	~300	P1a (赵义仑)
BrowserPipeTool	~400	P1a (赵义仑)
MAC Policy	~200	P1a (赵义仑)
P1b 业务支持		
SkillLoader	~300	P1b
CompositeMemory	~200	P1b
Critic	~100	P1b
P2 浏览器对接		
SgClawProcessHost	~250	P2
PipeListener	~150	P2
MacWhitelistCheck	~100	P2
P3 业务技能		
黄金样本 Skill	~150	P3
AI 翻译 400+ Skill	AI 生成	P3
P4 前端发布		
Side Panel UI	~100	P4
Skill Manager	~150	P4
总计	~2250	-

从零开发预估：~13000 行 → 基于 ZeroClaw 仅需 ~2250 行（减少 83%）

沟通机制

每日站会

- 时间：每日 10:00
- 时长：15 分钟
- 内容：昨日进展 / 今日计划 / 阻塞问题

联调时间

- P1a + P2：Day 3-5，重点打通 Pipe 通信（**关键路径**）
- P1a + P1b：Day 6-7，集成 AgentRuntime

- **P1b + P3**: Day 6-7, Skill 加载测试
- **P2 + P4**: Day 4-5, UI 集成测试
- **全员**: Day 8-9, E2E 场景验收

代码审查

- **P1a 审查 P2** 的 C++ 代码 (Pipe 协议一致性)
- **P2 审查 P1a** 的 Rust 代码 (错误处理)
- **P1b 审查 P3** 代码 (Skill 沙箱安全性)
- **P4 UI** 由产品经理审查 (UX)

文档协作

- **接口文档**: P1a + P2 共同维护 (Pipe 协议)
- **Skill 文档**: P3 编写, P1b 审核
- **部署文档**: P4 编写
- **用户手册**: 产品经理 + P4

风险与依赖

技术风险

风险	影响	缓解措施	负责人
Pipe 通信不稳定	极高 (阻塞路径)	增加重试机制、降级方案	P1a, P2
P1a 工作量过大	高	拆分为 P1a + P1b, 并行开发	项目经理
AI 翻译质量不稳定	中	提高黄金样本质量、人工抽检	P3
LLM 推理延迟过高	中	本地模型优化、缓存策略	P1b
Skill 沙箱逃逸	高	严格白名单、代码审计	P1b, P3
银河麒麟适配问题	中	提前在真机测试	P4

外部依赖

- **LLM API**: 需要确认内网本地模型 (Ollama + Qwen) 部署方案
- **测试环境**: 需要银河麒麟 V10 真机
- **业务系统访问权限**: P3 需要 OA/ERP 测试账号

关键路径

Day 1-2: 环境搭建 (所有人)

Day 3-5: Pipe 打通 (P1a + P2) ← **阻塞后续所有工作**

Day 6-7: 并行集成 (P1a+P1b, P1b+P3, P2+P4)

Day 8-9: E2E 测试 (所有人)

Day 10: 发布 (P4)

P1a 是关键路径，Day 5 前必须完成 Pipe 通信打通

交付清单

代码仓库

- gitlab.com/superrpa/sgclaw (Rust 引擎)
- gitlab.com/superrpa/chromium (C++ 集成, 分支 `feature/sgclaw`)
- gitlab.com/superrpa/agent-vue (Side Panel UI + Skill Manager)

文档

- ☒ 团队分工.md (本文档)
- ☐ API 接口文档.md (P1a + P2)
- ☐ Skill 开发指南.md (P3)
- ☐ Skill 翻译 Prompt 模板.txt (P3)
- ☐ 部署手册.md (P4)
- ☐ 测试报告.md (P4)
- ☐ 用户手册.md (产品 + P4)

安装包

- ☐ sgclaw-v1.0.0-kylin-v10-amd64.deb
- ☐ sgclaw-v1.0.0-windows-x64.exe

演示材料

- ☐ 演示视频 (6 大业务场景)
 - ☒ PPT 宣讲材料 (已完成)
-

附录：工具链

P1a 工具链

```
# Rust 编译环境
rustup default stable
rustup target add x86_64-unknown-linux-musl
rustup target add x86_64-pc-windows-msvc

# 依赖管理
cargo install cargo-edit
cargo install cargo-audit

# 测试
cargo test
cargo clippy
```

P1b 工具链

```
# 同 P1a Rust 环境

# SQLite 工具
sudo apt install sqlite3 libsqlite3-dev

# JS 运行时 (二选一)
# rusty_v8 或 boa_engine, 通过 Cargo.toml 依赖
```

P2 工具链

```
# Chromium 编译环境
depot_tools/
gn
ninja

# 代码格式化
clang-format

# 测试
out/Default/unit_tests --gtest_filter=SgClaw*
```

P3 工具链

```
# Node.js 环境
nvm install 18
npm install -g ed25519-cli

# Skill 签名
node sign-skill.js <skill-file>

# Skill 验证
node verify-skill.js <skill-file>

# AI 翻译 (内网大模型)
# 访问内网 Qwen-72B / DeepSeek API
```

P4 工具链

```
# Vue 开发
npm install
npm run dev
npm run build
```

```
# 测试
npm run test:unit
npm run test:e2e

# 打包
./scripts/package-deb.sh
./scripts/package-exe.sh
```

团队人员配置建议

技能要求	P1a (赵义仑)	P1b	P2	P3	P4
Rust	✓✓✓	✓✓✓	-	-	-
C++ / Chromium	-	-	✓✓✓	-	-
业务理解	-	-	-	✓✓✓	-
提示词工程	-	-	-	✓✓	-
Vue / 前端	-	-	-	-	✓✓✓
DevOps	-	-	-	-	✓✓
工作优先级	P0	P1	P0	P1	P2

推荐招聘策略：

- P1a (赵义仑)：项目核心，负责关键路径
- P1b：招聘 Rust 工程师，有 LLM Agent 经验优先
- P2：公司内部调配 C++ 浏览器工程师
- P3：招聘熟悉业务的 RPA 工程师 + 提示词工程经验
- P4：公司内部调配前端工程师 + DevOps

文档版本：v2.0 最后更新：2026-03-04 维护者：项目经理 重要变更：

- 团队从 4 人扩展为 5 人
- P1 拆分为 P1a（核心通信）和 P1b（业务支持）
- P3 工作方式改为 AI 辅助迁移 400+ 场景
- P4 增加 Skill 管理后台开发
- 代码量从 3100 行调整为 2250 行